Introduction to the EO Exploitation Platform Common Architecture (EOEPCA)

living planet symposium | MILAN 13–17 May 2019

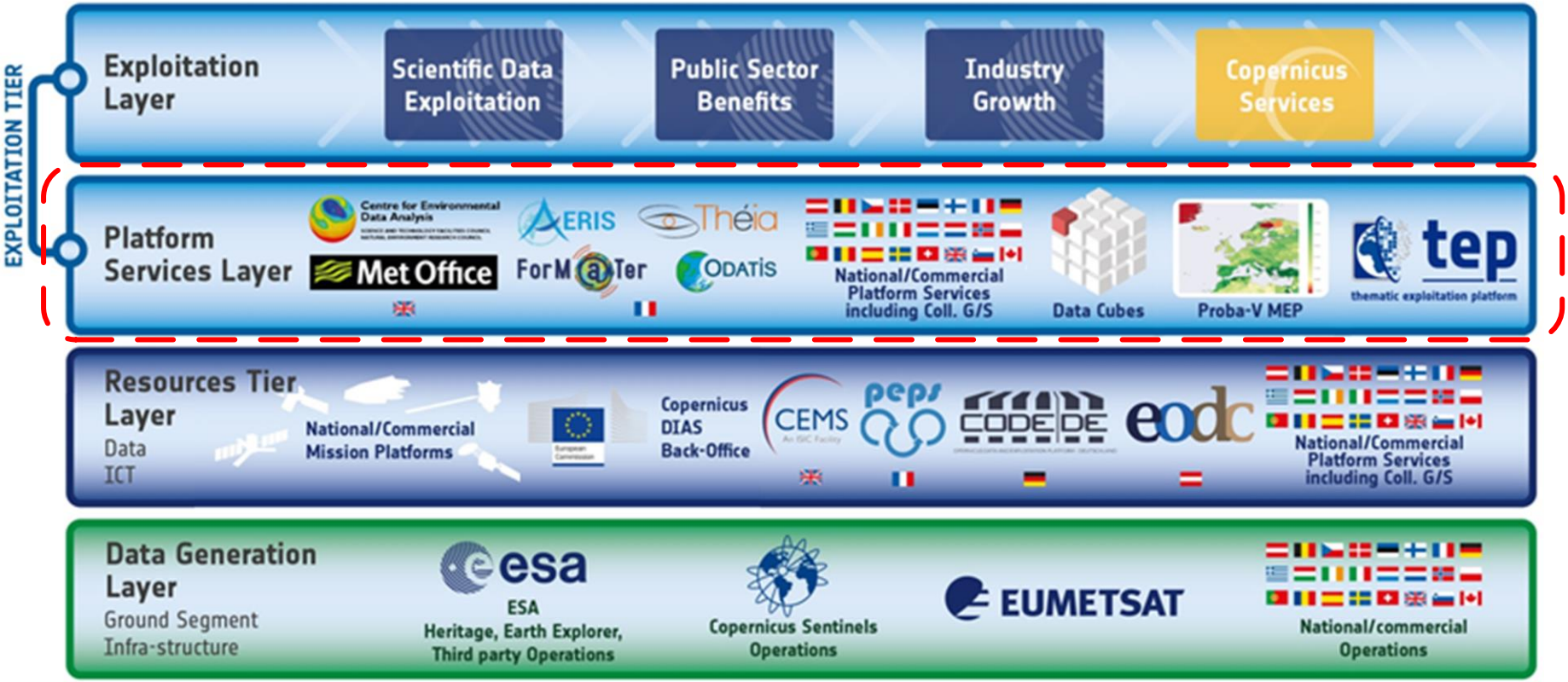
Telespazio VEGA UK
16th May 2019

Introducing EOEPCA

EOEPCA intends to become a widely adopted reference architecture and implementation for the future exploitation of distributed EO data and services.

- Open design, freely available for reuse on GitHub
- Interfaces/data based on Open Standards to facilitate interoperability amongst collaborating platforms
- Design process to be consensus based to be as inclusive as possible

EOEPCA Context



Common Architecture Benefits

Interoperability

- Facilitated by use of Open Standards for interfaces/data
- Ability to access datasets from other domains
- Possibility to build workflows that bring in processing at other facilities

End-users

- Benefit from a consistent experience

Experts

- Benefit with ‘portable’ services across platforms

Platform providers

- Can more easily integrate with other similar platforms
- Re-use of Reference Implementation as building blocks

Process

Engage
Analyse/evolve use cases
Identify/consult stakeholders
Capture feedback

Design Architecture
Identity/analyse activities
Define building blocks (MSDD and MICD)

Explore & Innovate
Testbeds to explore use cases
Pilots to consolidate architecture

Implement
Select domain area experts
Refine MSDD and MICD
Reference Implementation

Operate
Demonstrate architecture
Collect user feedback

Consolidate
Validate use cases
Standards Recommendations

Domain Area Experts procured to perform detailed design and develop **Reference Implementation**

Community Feedback

Community feedback is sought through review/comment of project documentation…

- Use Case Analysis: https://eoepca.github.io/use-case-analysis/
- Master System Design: https://eoepca.github.io/master-system-design/

Documents currently in-progress, but first Draft soon to be issued

Raise comments as Issues in GitHub or email the team:
Eoepca.SystemTeam@telespazio.com

Requirements for system design

USE CASES

Use Cases

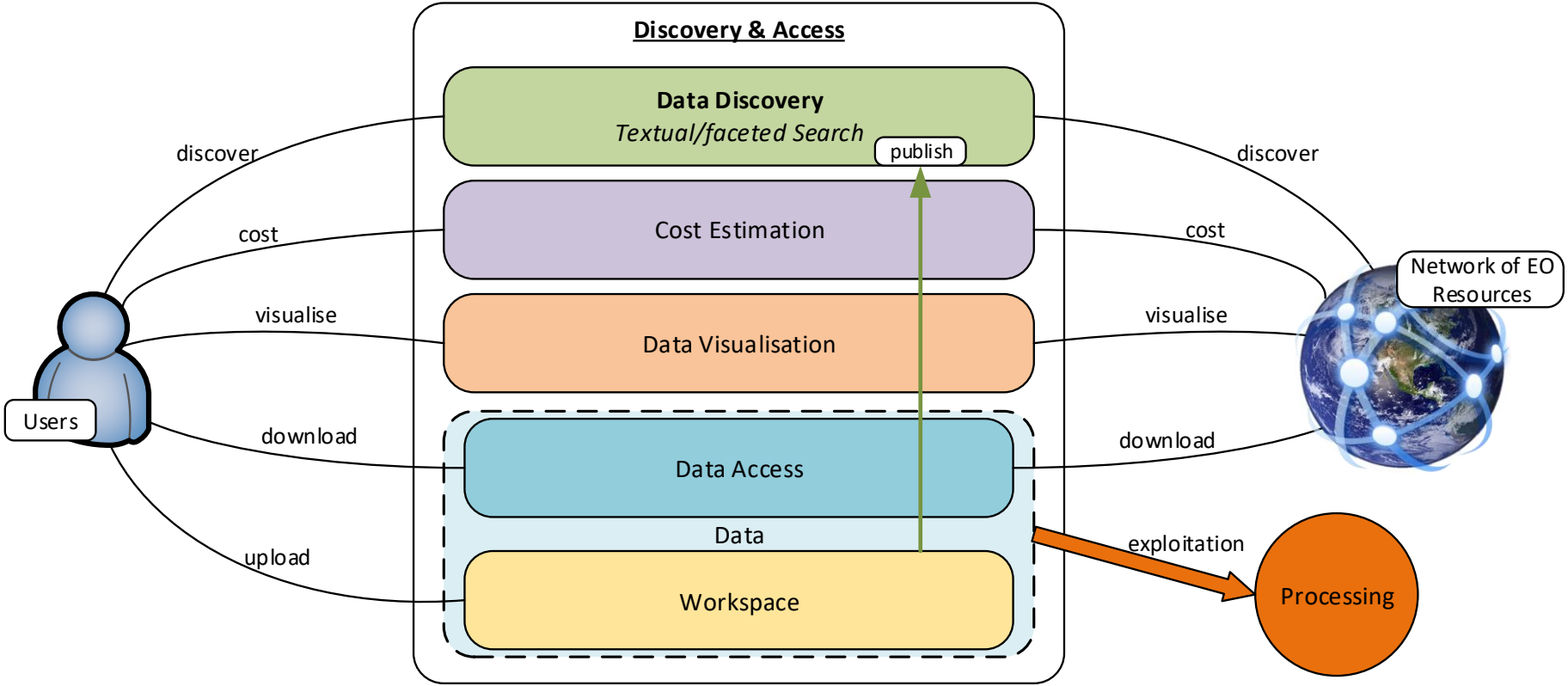
Consumer:

- discovers and visualises value-added products
- uploads data into its own workspace
- discovers and executes processing services
- discovers and executes interactive applications
- analyses value-added products
- executes bulk processing
- executes systematic processing
- performs open science
- accesses the EP with an external application

Expert user:

- builds new processing services
- builds new processing services chains
- builds new interactive applications
- builds new value-added products
- develops and executes code via a web development environment
- performs training

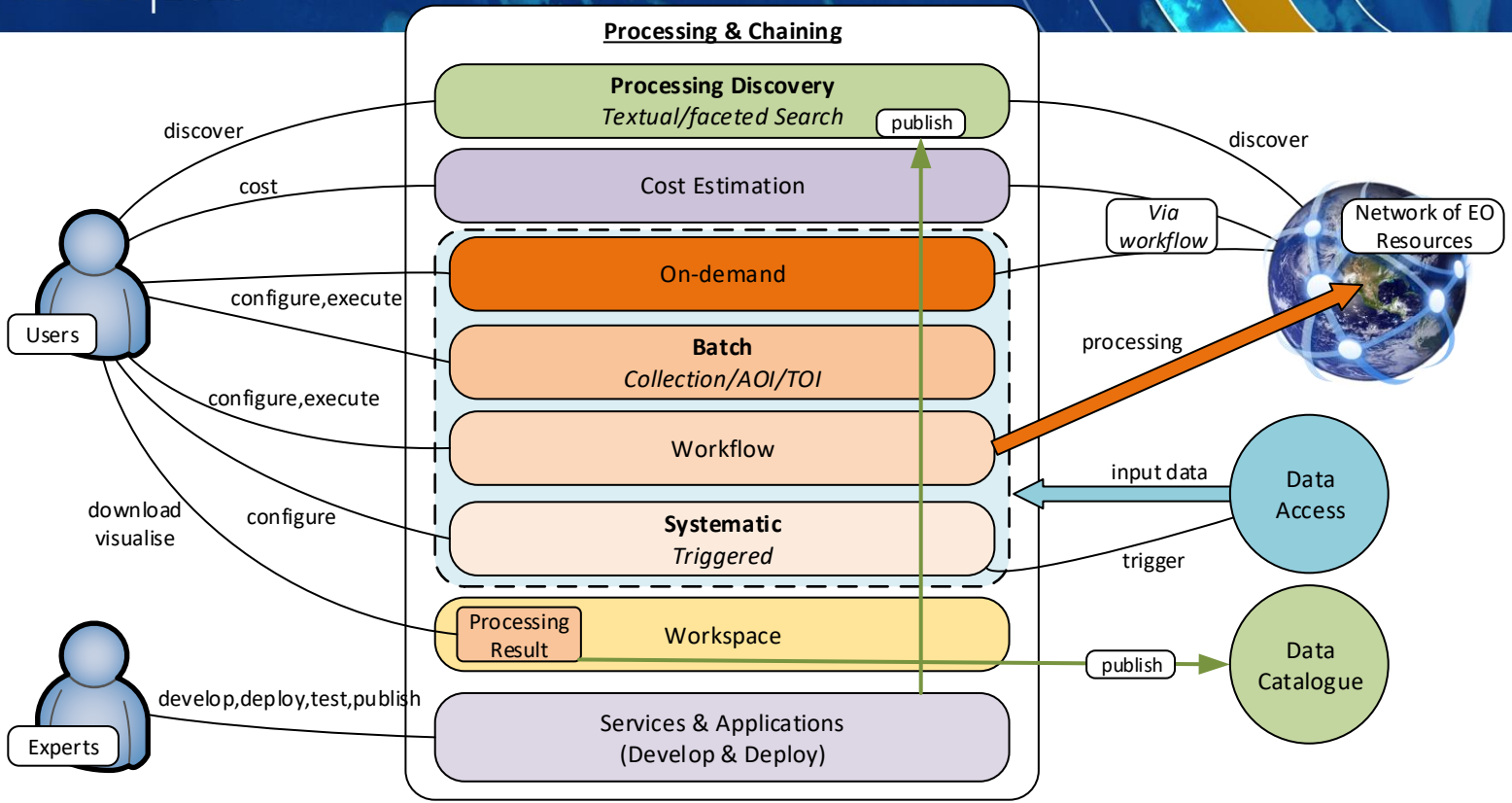
Use Cases: Data Access



Use Cases: Data Access

- Guest (anonymous) users may be permitted to discover data
- Users must be logged in to exploit data
- Access controls include:
    - Accepted terms and conditions of use
    - Have appropriate permissions
    - Have enough ‘credits’ to cover usage
- Discovered data can be visualised
- Own data can be uploaded and visualised
- Uploaded data can be published (with control over permissions)
- Data is discoverable and accessible from other platforms (federation)

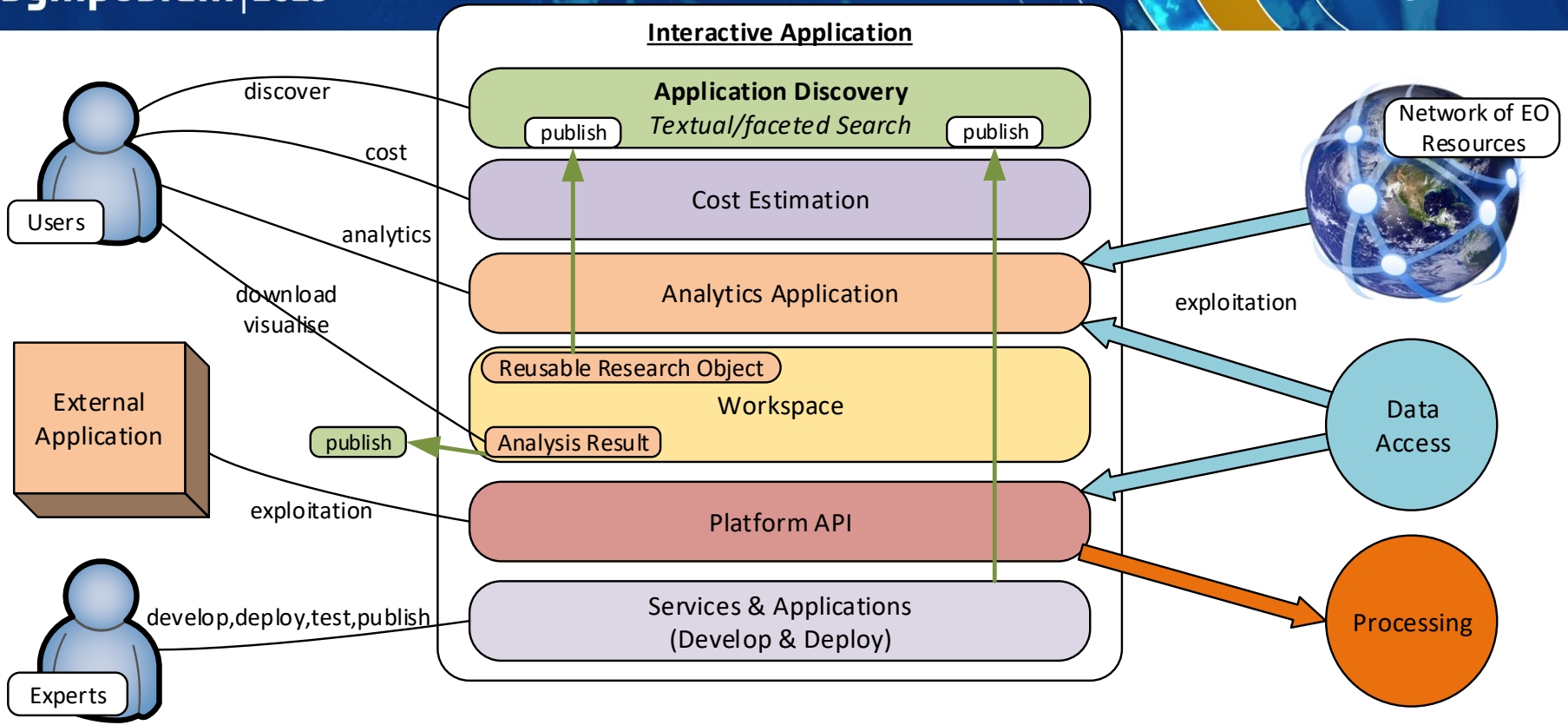
Use Cases: Processing



Use Cases: Processing

- Guest (anonymous) users may be permitted to discover processing services
- Users must be logged in to execute processing
- Access controls include:
  - Accepted terms and conditions of use
  - Have appropriate permissions
  - Have enough ‘credits’ to cover usage
- **On-demand:** ad-hoc single execution processing
- **Batch:** bulk data processing specified by criteria (e.g. Collection/AOI/TOI)
- **Systematic:** routine processing driven by trigger event (e.g. time schedule, or new data arrival)
- **Workflow:** pipeline (DAG) of chained processing steps
- Workflow steps can be invoked on other platforms (close to the data) *Federated discovery and exploitation of data/processing on other platforms*
- Processing Results can be published (with control over permissions)
- Experts develop, deploy and publish new processing services (with control over permissions)

Use Cases: Applications



Use Cases: Applications

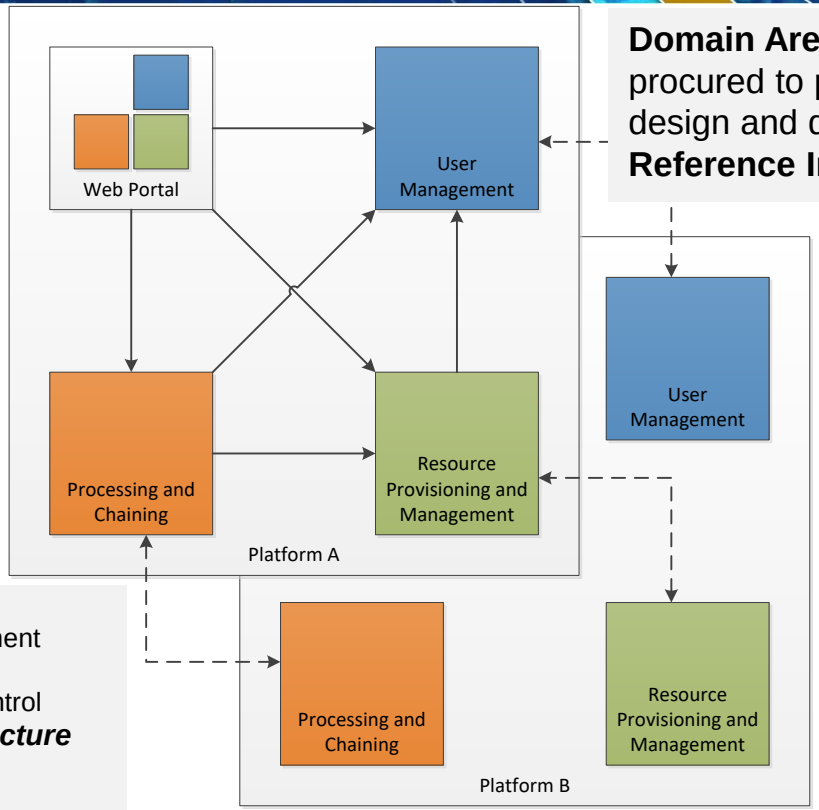
- Guest (anonymous) users may be permitted to discover applications
- Users must be logged in to perform analysis
- Access controls include:
    - Accepted terms and conditions of use
    - Have appropriate permissions
    - Have enough ‘credits’ to cover usage
- Web Applications: delivered through a ‘native’ browser interface
- Remote Applications: delivered through remote desktop via browser
- Analysis can include data in other platforms (federation)
- Analysis Outputs can be published (with control over permissions)
- Analysis can be captured as Reusable Research Objects (recording data used, processing performed, user notes). Can be published (with control over permissions).
- External Applications: use the *Platform API* to access platform services
- Experts develop, deploy and publish new applications (with control over permissions)

Building blocks and interfaces

SYSTEM ARCHITECTURE

Architecture: Domain Areas

The System Architecture must provide a solution stack that meets the needs of an EO Exploitation Platform in delivering its individual service provision, and at the same time must provide an architecture that allows each such deployment to share its services within the federation of EO Exploitation Platforms



Domain Area Experts
procured to perform detailed design and develop
Reference Implementation

Design
Master System Design Document / <component> Design Document
Interfaces
Master Interface Control Document / <component> Interface Control
Community participation in the review of the architecture

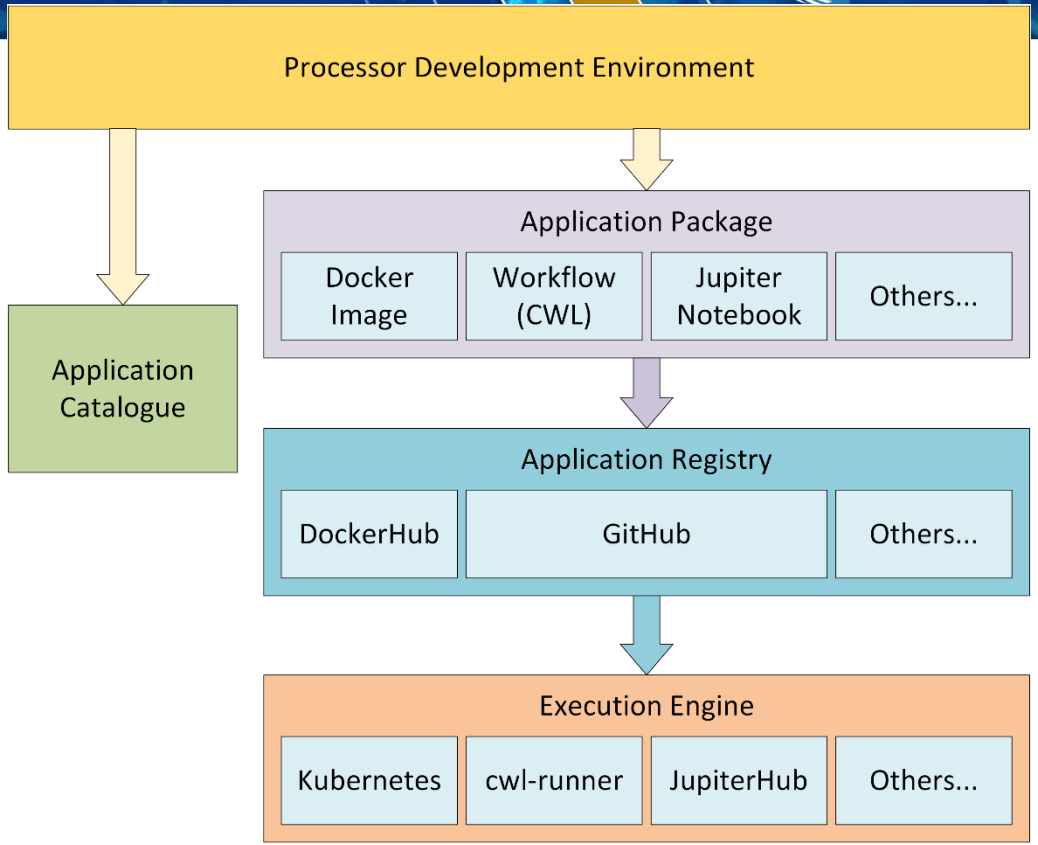
PROCESSING & CHAINING

Processing & Chaining

Provide an extensible repository of processing functions, tools and applications that can be discovered by search query, invoked individually, and utilised in workflows (cross-platform)

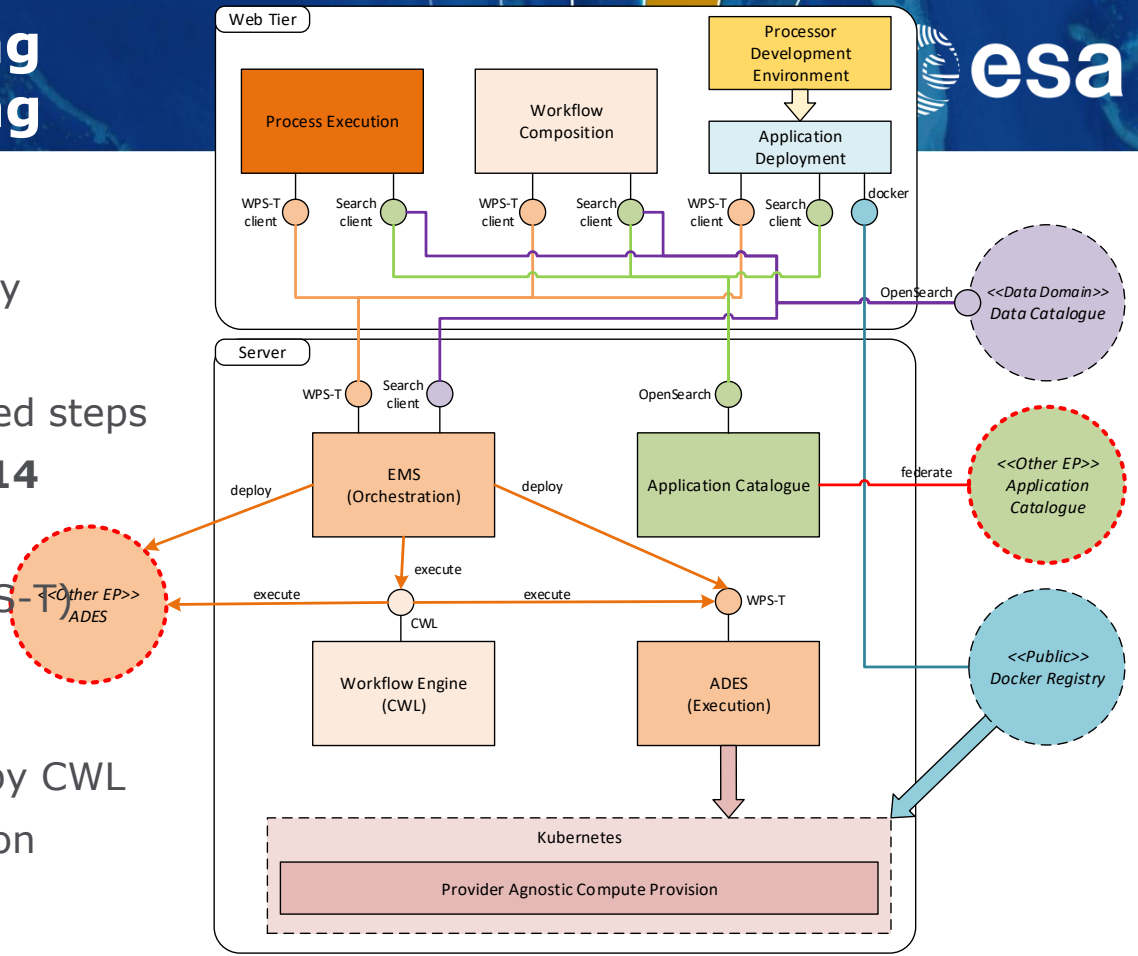
General Concept

- Discovery
  - Application Catalogue
- Extensible
  - Artefact Registry
  - Packaging Format
- Execution Engine
  - Processing services
  - Workflows
  - Analytics Applications
  - Interactive Applications



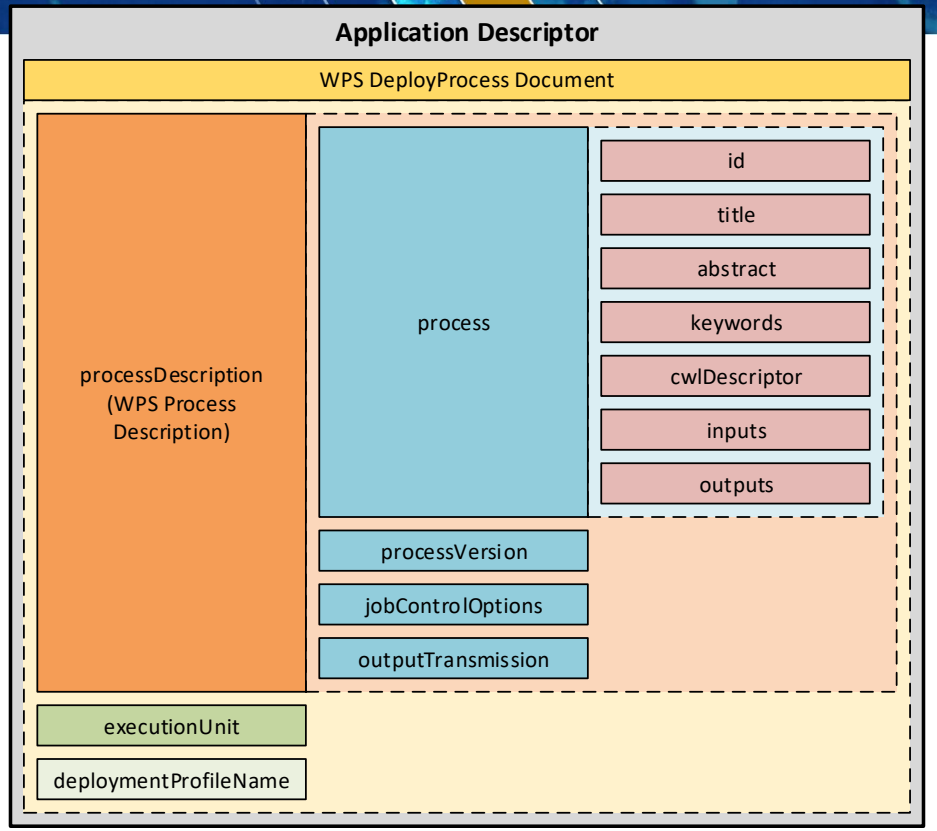
Processing & Chaining

- Challenges:
  - Data/service-IO interoperability
  - Platform independence
  - Access controls across federated steps
- **Based upon OGC Testbeds 13/14**
- Packaged as Docker images
- EMS: Workflow Orchestration (WPS-T)
  Local and remote-EP steps
- ADES: Process execution (WPS-T)
- Processor input/output described by CWL
- Kubernetes resource tier abstraction

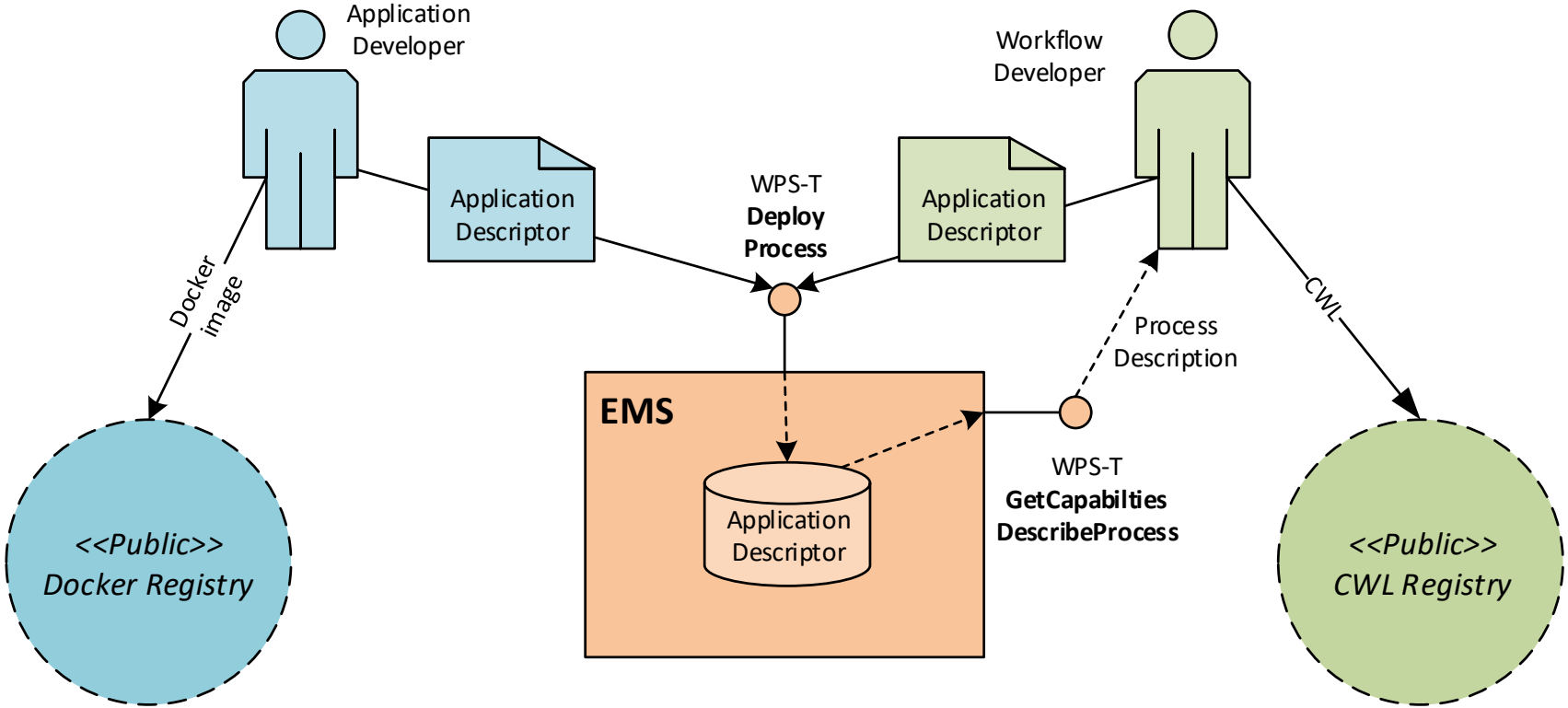


Application Package

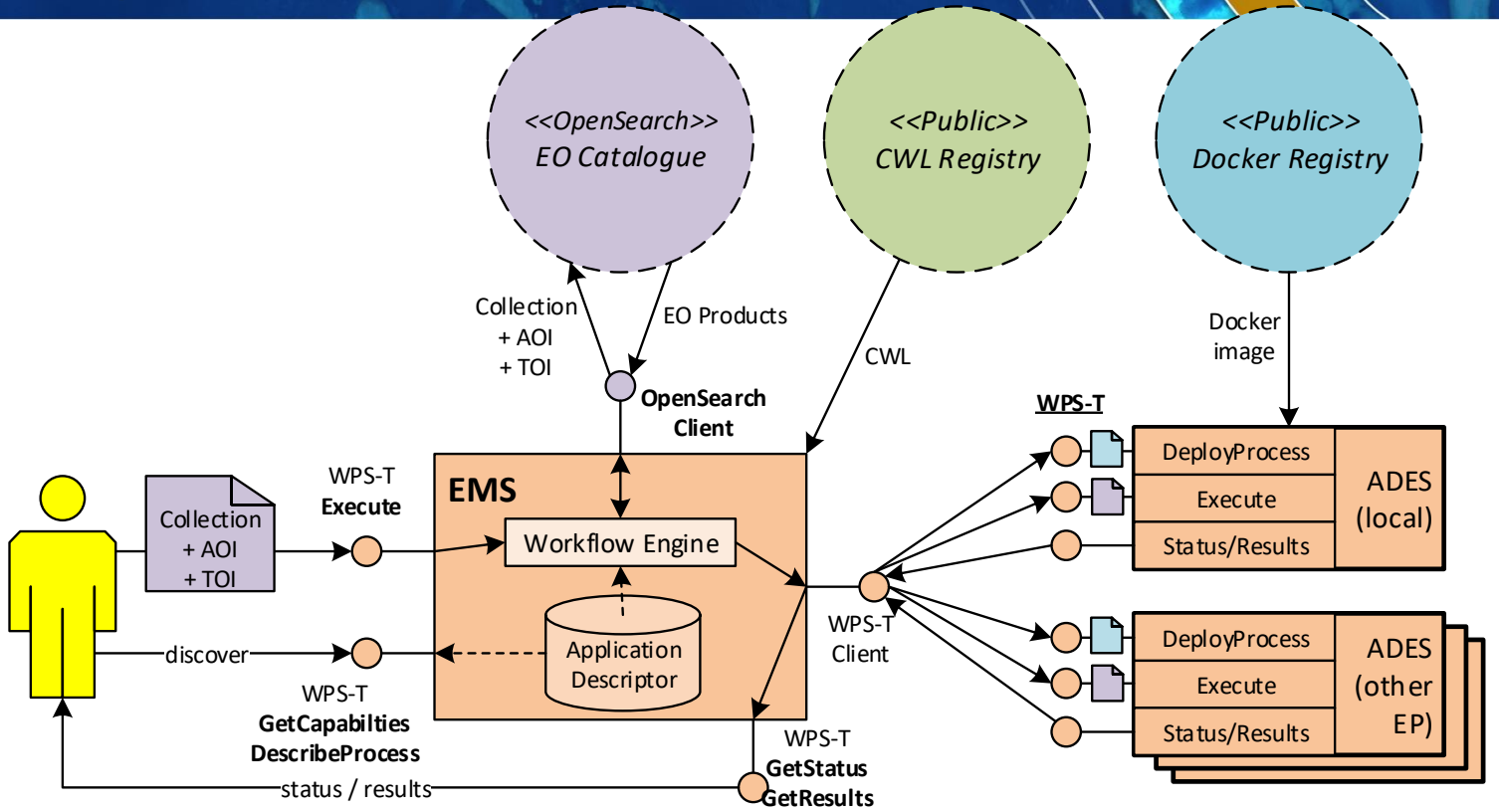
- Platform independent and self-contained representation of a software item
- The Application Package comprises two main parts:
  - Application Descriptor - metadata descriptor file
    - A link to the application artefact (execution unit)
    - A description of the application's inputs and outputs
    - Other auxiliary information
  - Application Artefact – i.e. the 'software' component that represents the execution unit
- cwlDescriptor formalises input/output definition
- executionUnit:
  - dockerizedApplication
    URL of the docker image to run
  - Workflow
    URL of the CWL file that defines the workflow



Develop / Deploy

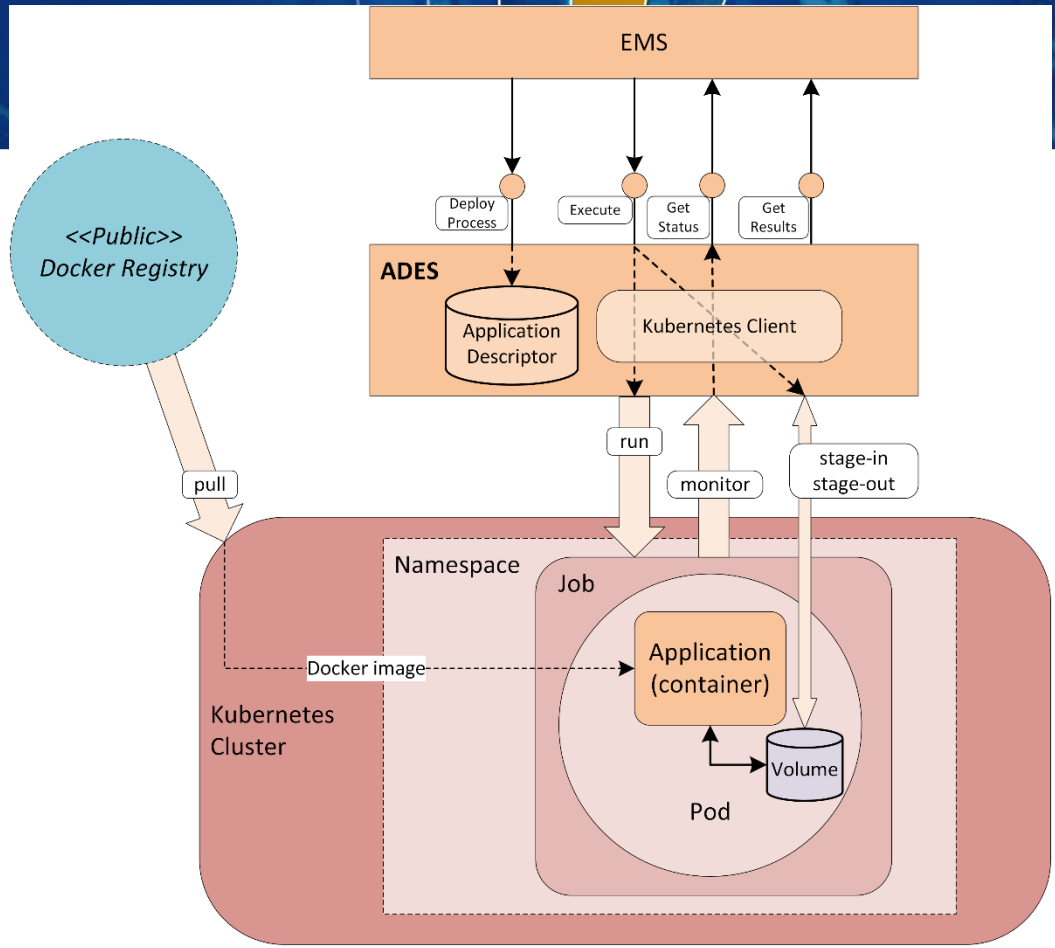


Workflow Execution



Process Execution

- EMS deploys service to WPS
- K8s Job provides platform agnostic container execution
- K8s Namespaces for the purpose of establishing a sandboxed execution environment for each task execution
- Data stage-in/out:
  - Local mounted volume
  - S3, Swift, OGC-Wxx
  - Data Access Library
  - FUSE file-system ?



WPS-T REST/JSON

WPS-T REST/JSON

- Work in-progress
- OGC WPS 2.0 SWG

Not shown: interfaces for quotation/billing

Resource	HTTP Method	Description	WPS operation
/	GET	The landing page provides links to the API definition, the Conformance statements and the metadata about the processes offered by this API	
/processes	GET	Retrieve available processes	GetCapabilities
/processes	POST	Deploy a process	DeployProcess
/processes/{id}	GET	Retrieve a process description	DescribeProcess
/processes/{id}	DELETE	Undeploy a process	UndeployProcess
/processes/{id}/jobs	GET	Retrieve the list of jobs for a process	
/processes/{id}/jobs	POST	Execute a process	Execute
/processes/{id}/jobs/{jobID}	GET	Retrieve the status of a job	GetStatus
/processes/{id}/jobs/{jobID}	DELETE	Dismiss a job	
/processes/{id}/jobs/{jobID}/result	GET	Retrieve the result(s) of a job	GetResult

Application Catalogue

- **Discovery:** Provides an inventory of processing services that acts as a Marketplace for the discovery and browse for processing services
- **Metadata:** To understand what data an application can be applied to, and how to chain in a workflow – ***overlap with ApplicationDescriptor***
- **Current OGC Testbed-15** (EOPAD)
  The Application Catalogue is the subject of the current OGC Testbed-15 through which the Data Model and catalogue Service Interface are being explored:
  - **OGC 17-084** (GeoJSON(-LD) metadata encoding for EO collections)
    Explore the capabilities of OGC 17-084 to encode application metadata
  - **OGC 17-047** (OGC OpenSearch-EO GeoJSON(-LD) Response Encoding Standard)
    Explore the capabilities of OGC 17-047 to encode OpenSearch responses in GeoJSON(-LD)
  - Use of multi-step discovery and faceted search
  - **Registration:** Explore transactional extension to **OGC-CSW** for application registration

Other Applications

Interactive Applications

- WPS does not facilitate the invocation of GUI-based interactive applications which offer a synchronous experience to the end-user
- Dockerized approach **does** lend itself to the packaging of GUI-based applications:
  - **Web applications**
    Runs as a self-contained service within the container. The web application is delivered through the portal interface of the hosting exploitation platform
  - **Native applications**
    A remote desktop (RDP) approach is used to present the interface to the user, typically rendered through a web page presented in the user's browser. For example, using Apache Guacamole.

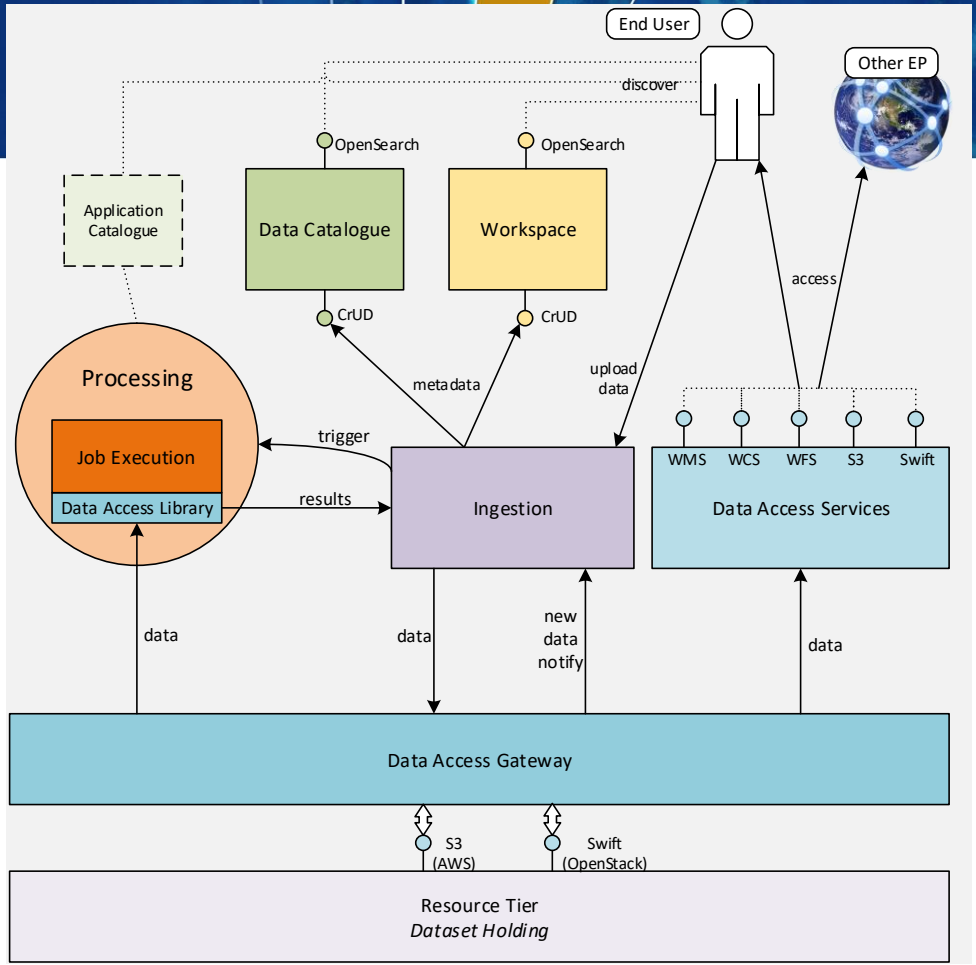
Parallel Processing

- OGC Testbeds 13/14 only consider serial processing jobs running in a single Docker container
- Possible approach within framework…
  Dockerized processor makes subordinate invocations that exploit some specific data processing clustering infrastructure available within the platform.
- For example, the invoked process executes some Python code that then invokes a dask or SLURM cluster to perform the processing work
- Dependency on the existence of the expected cluster within the platform:
  - Processing service declares within its Application Deployment Package, that it 'requires' a particular service, e.g. dask
  - EP declares within its WPS capabilities document, that it 'provides' particular services

RESOURCE MANAGEMENT

Resource Management

- Resource Discovery:
  - **Data Catalogue**
  - Discovery of federated resources
  - *Related: Application Catalogue (see Processing & Chaining)*
- **Data Access** through common protocols (open standards)
  Access from users (download/visualise), processing services/applications, and other EPs
  - OGC Web Map Service (WMS)
  - OGC Web Coverage Service (WCS)
  - OGC Web Feature Service (WFS)
  - Services provided by Resource Tier:
    - AWS S3 Object Store
    - Swift Object Store (OpenStack)
- **Abstraction** of data access interface to the infrastructure Resource Tier
- **Ingestion** of contributed data/results
- Workspace for user and group collaboration



Data Catalogue

- Provides the user the capability to discover data/products by browse/search, and to obtain details on specific data/products discovered
- Data organised hierarchically:
  - **Browse** metadata (collection)
    ISO19115 records – high level information (title, description, coverage, access rights, T&Cs)
  - **Discovery** metadata (product)
    Defined for each granule (file) comprising the collection. Includes spatial/temporal coverage, variable, file-type, access method(s)
  - **Archive** metadata (file)
    File header information. Used to seed Discovery data.
- **Search** – EO OpenSearch
  - Extensions: Geo/Time (10-032r8), EO (13-026r8)
  - Encodings: GeoJSON(-LD) Metadata (17-003r1/17-084), GeoJSON(-LD) Response (17-047)
  - CEOS OpenSearch Best Practice (CEOS-OPENSEARCH-BP-V1.2)
- **Management** (internal CrUD)
  Supporting add, update and delete operations for metadata records.
  ??? OGC-CSW with Transactional extensions, or interface such as ElasticSearch API ???

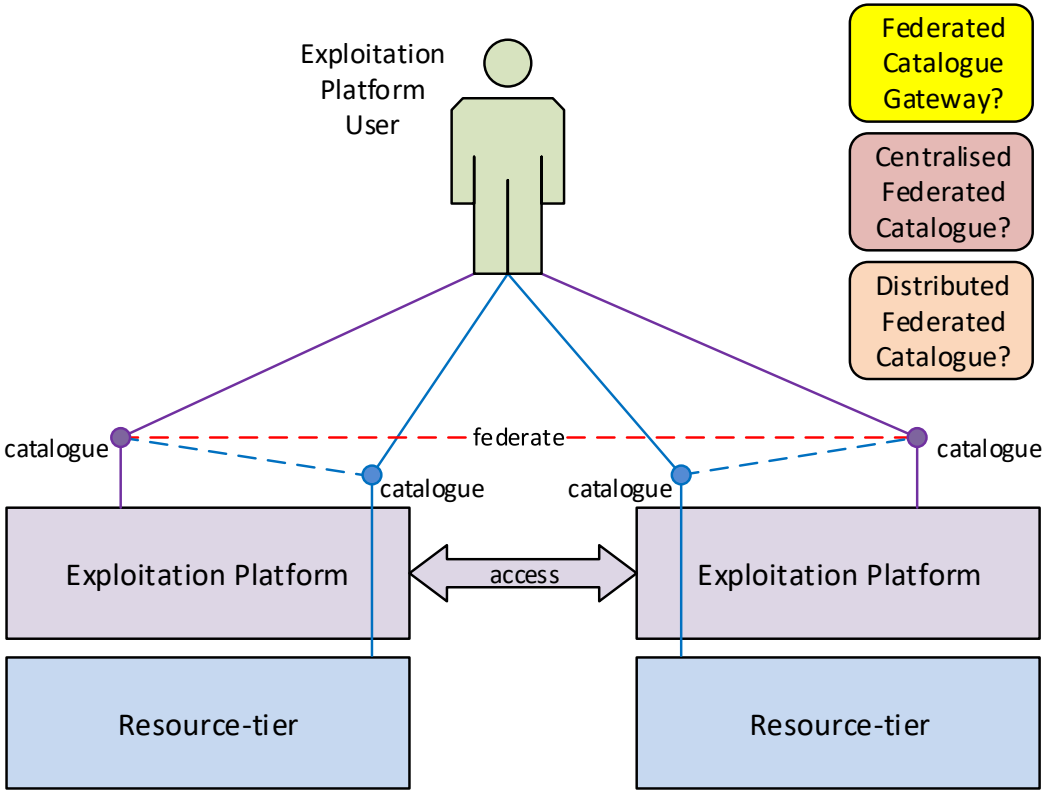
Federated Discovery

How does a user find the data/services they need in a federated system of Exploitation Platforms ?

- Federated access between Exploitation Platforms
- Users must discover the data/services available between EPs
- Catalogues at multiple layers: Resource Tier / Exploitation Tier

Possible approaches?

- Gateway – A central proxy
- Centralised – Central mirror
- Distributed - Catalogues mirror each other



Data Access

- EO datasets are stored according to the underlying storage technology of the infrastructure Resource Tier. EP interfaces:
  - **Processing services and applications: stage-in/out of data/results**
  - Platform Data Access Services (WMS,WCS,etc.): access to datasets
  - Ingestion: storage of ingested data
- Resource Tier interfaces: AWS S3, Swift Object Store (OpenStack), Others?

- Data Access Library (for processing services and applications)
  - Common Data Model for library language bindings, e.g. python, javascript
  - Platform independence: Alternative deployment-specific implementations can be 'plugged-in' to processing services and applications. Handled by the execution engine at runtime
- Data Access Gateway
  - Abstraction layer on top of the underlying storage
  - Present a well-defined storage interface to the other components of the Exploitation Platform
  - Provides a common data access interface to the other components of the Exploitation Platform that they can target in their implementation
  - What interface to present ?
  - Possibility to lose functionality in the abstraction ?

Ingestion / Workspace

Ingestion

- Presents a standard interface to the EP components, whilst transparently interfacing with the infrastructure Resource Tier
- QUESTION: What standard interface does it present ?
- Performs the following steps:
  - Authorisation check
  - Quota check
  - Metadata extraction
  - Preview generation
  - Format conversion
  - Storage PUT
  - Catalogue PUT
  - Trigger notifications
- Notifications:
  - Raise indicators to users (visual, emails, etc.)
  - Trigger systematic actions in other EP services (e.g. systematic processing)

Workspace

- Used by users to organise data/processing-services that are of current interest to them, they are currently working on, and to organise results of processing executed, Research Objects, etc.
- Concept can be extended to create a Group Workspace for sharing and collaboration
- Can be modelled as a Catalogue – in which the browse/discover access privilege is limited either to an individual user (personal workspace) or a group of collaborating users (group workspace)
- OpenSearch for read
- TBD interface for add, update, delete

USER MANAGEMENT

User Management

Identity and Access Management (IAM)

Identification/authentication of users and authorization of access to protected resources (data/services) within the EP.

Accounting and Billing

Maintaining an accounting record of all user accesses to data/services/applications, supported by appropriate systems of credits and billing.

User Profile

Maintenance of details associated to the user that may be needed in support of access management and billing.

Identity & Access Management

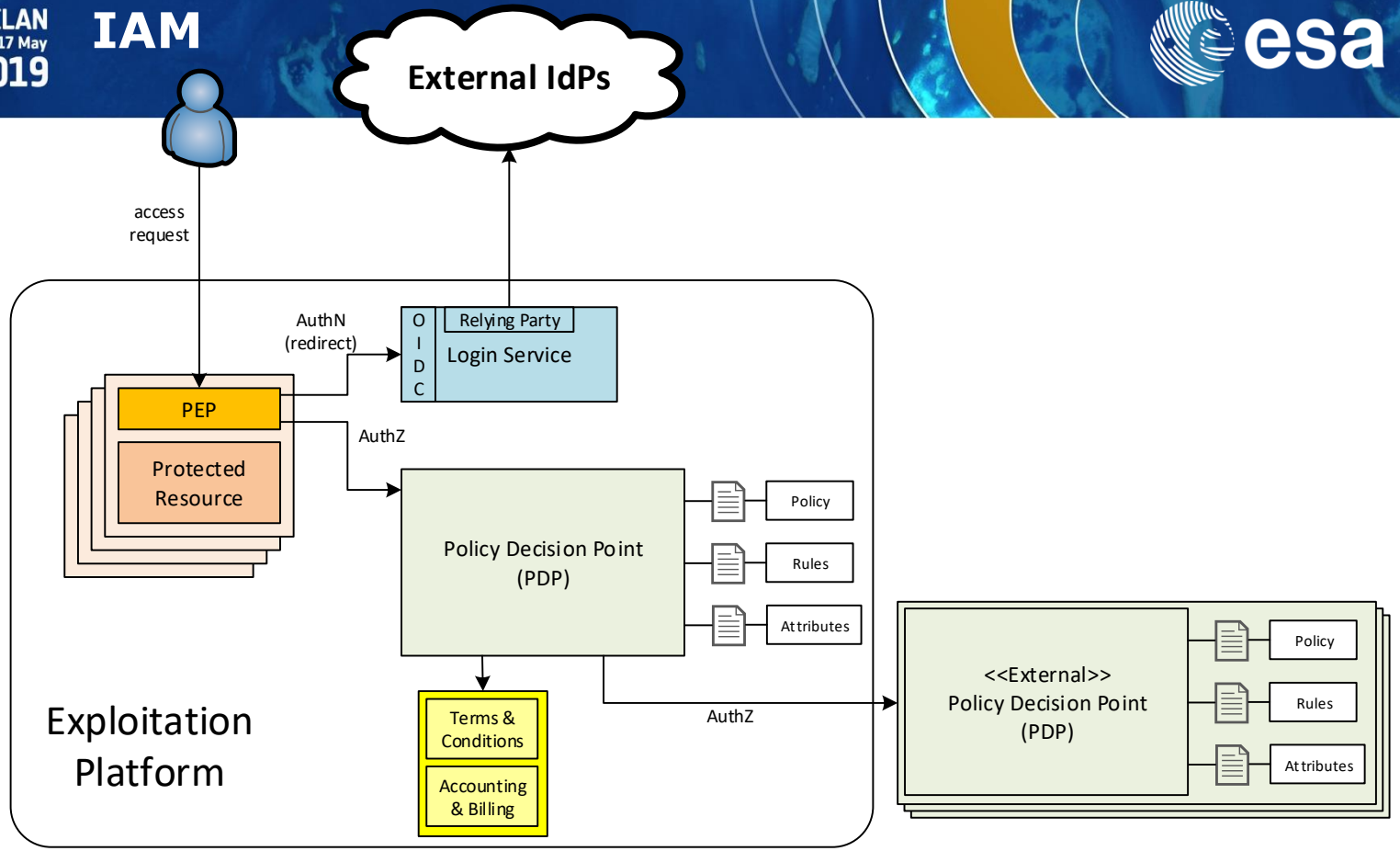
The goal of IAM is to uniquely identify the user and limit access to protected resources to those having suitable access rights

- **Bring You Own Identity:** users authenticate to the EP with their existing external identity
- Attribute Based Access Control (ABAC): access policies/rules that define attributes required by resources and possessed (as claims) by users:
  1. Unique user identification
  2. Determine the access policy applicable to the resource
  3. From the access policy determine: what attributes are required to access the protected resource; whether the user has the required attributes
- Separation of concerns (Identification (1) from Authorisation (2/3)):
  - Federated Identity is provided externally – 'home' IdPs do not define access policy
  - Authorisation policy is enforced within the platform at point of access
  - Authorisation policy definition is federated within the network of EO resources

IAM

IAM Approach

- **PEP** (Policy Enforcement) as an incoming request 'filter'
- Authentication **delegated** to External IdPs
- **PDP** (Policy Decision) can be aggregated through ***federated authorisation***



Authenticated Identity

Login Service

An OpenID Connect Provider that provides a 'Login With' service that allows the platform to support multiple external identity providers. Dual role:

- OIDC Provider – for interface with EP clients
- Relying Party – for interface with external IdPs

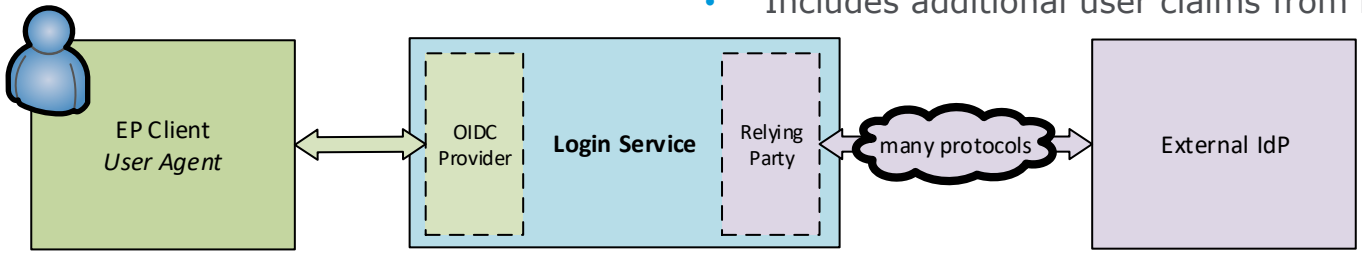
From the perspective of the EP Clients this is an OIDC flow

OIDC Provider

- Presents an OIDC Provider interface to its clients, through which the OIDC clients can obtain Access Tokens to resources
- The access tokens are presented as ***Bearer Tokens*** by the clients in their requests to resource servers (intercepted by PEP)

ID Token

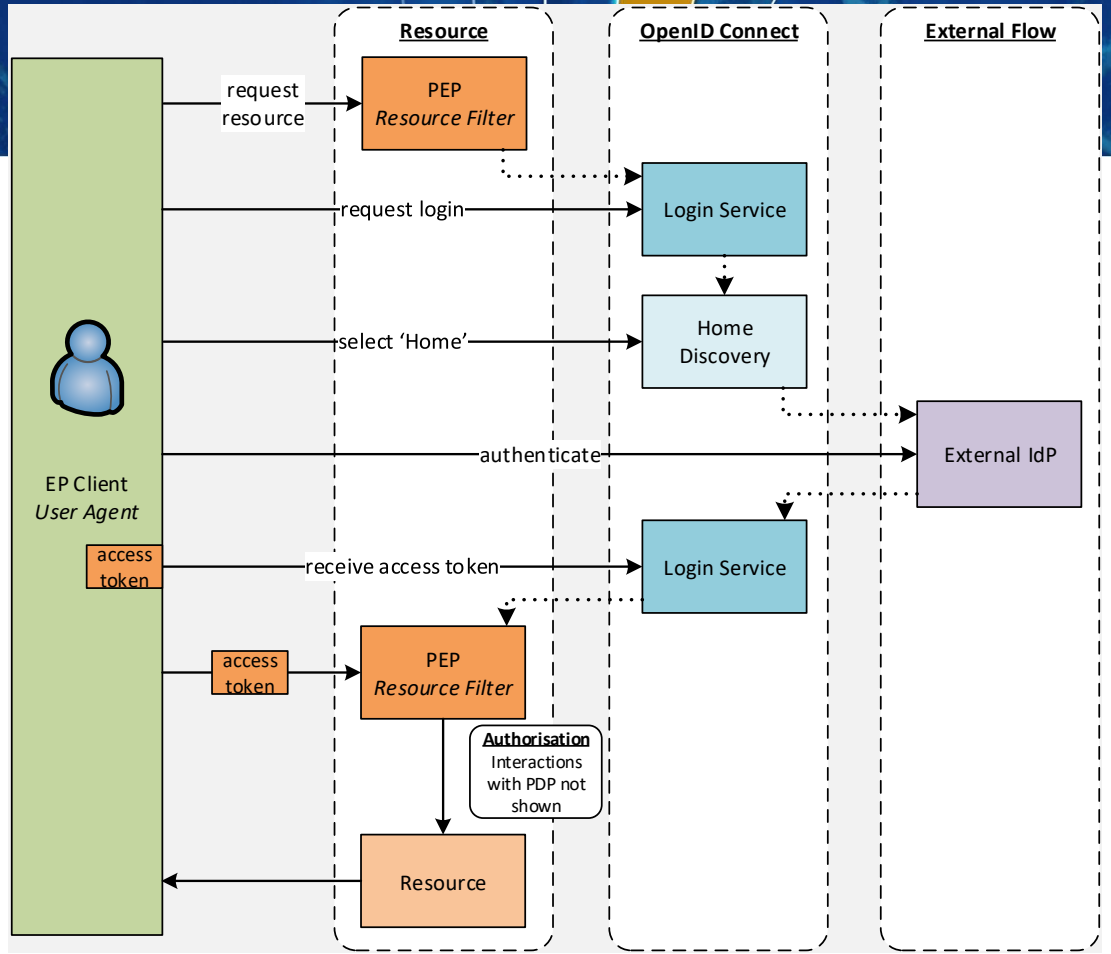
- Signed JWT (JWS) returned from the /userinfo endpoint of the Login Service
- Asserts a user's authenticated identity with integrity
- Includes additional user claims from External IdP



AuthN Flow

Relying Party

- Login Service requests as a Relying Party delegated access to the user's identity from External IdP
- External IdPs: EduGain, GitHub, Google, Twitter, Facebook, LinkedIn, others TBD
- Login Service must establish trust relationships (e.g. Client Credentials) with External IdPs
- The interface/flow with the External IdP is integrated into the OIDC flow implemented by the Login Service
- The interactions with the external IdP represents the 'user authentication step' within the OIDC flows
- Completion of a successful authentication with the external IdP allows the Login Service to issue the requested access tokens



Authorisation

- The PEP (acting on behalf of the Resource) relies upon the access token to establish the authenticated identity of the users making the requests (ID Token)
- Once the user identity is established, then the PEP can continue with its policy decision (deferred to the PDP)
- PEP interrogates the PDP for an authorisation decision, containing:
  - Identity of end-user (subject)
  - The API (path/version etc.) being accessed (resource)
  - The operation (HTTP verb) being performed (action)
- PDP returns an authorisation decision based upon details provided in the request, and the applicable policy
- Policy decision may depend upon dynamic 'attributes', such as whether the user has enough credits to 'pay' for their usage, or whether they have accepted the necessary Terms & Conditions for the resource
  - PDP must interrogate other EP-services such as 'Accounting & Billing' and 'User Profile' to answer such questions
- PEP <-> PDP Interface
  Interface TBD. Possibilities include:
  - XACML, e.g. with REST/JSON profile
  - OAuth UMA
- Federated Authorisation (PDP <-> Other PDP)
  - The authorisation policy may delegate all or part of the decision to external PDP(s) within the federated network
  - Represents a Federated Authorisation model that facilitates shared resources and virtual organisations
  - Use same interface technology as for PEP<->PDP

Federated Access

Typical example: Workflow execution where one or more steps are conducted on other platforms.
Remote execution (M2M) must be conducted in the user context (delegated) of the originating request

Possible solutions...

- **User Pre-authorisation**
  - User conducts OIDC flows from one EP (Relying Party) to another EP
  - Originating EP obtains delegated access (access/refresh tokens) to another EP on behalf of the user
  - Requires establishment of many-to-many trust relationships
- **Trusted Identity Assertion**
  - OIDC JWKS provides a distributed key-hierarchy – chain of trust
  - Public keys dynamically shared to underpin the signing and validation of JWTs
  - Use ID Token (JWT with short expiry) of originating user to assert user identity in remote EP
  - ID Token acts as a short-lived credential, and is carried through calls into and across resource servers

Other approaches ?

User Profile / Billing

User Profile

The User Profile is a system resource that maintains a set of data for each user including:

- User details
- Terms and conditions accepted by the user
- License keys held by the user
- User API key management

The User Profile for a given user is tied to the unique identifier provided by their Home-IdP through the authentication process

PDP interrogates User Profile for policy decision (attribute query)

Standards-based Interface for User Profile?

Quotation / Billing

Specific to each platform – but some hooks can be provided in support of a platform solution

Ref. OGC Testbed-14 ER [OGC 18-057]

- WPS-T (proposed) includes endpoints for:
  - Requesting quotation
    Estimate of processing cost before use
  - Executing job from prior quotation
  - Associating a bill to an executed job
- EMS/ADES can support these methods

CONCLUSION

In Conclusion

ITT: Domain Experts / Operator

- Elaborate system design and domain detailed design
- Incremental Reference Implementation development
- Deployment as operational service

Project resources…

- Project portal: https://earsc-portal.eu/display/EOEPCA
- GitHub: https://github.com/EOEPCA

Documents (currently in-progress) for review/comment…

Raise comments as Issues in GitHub

- Use Case Analysis: https://eoepca.github.io/use-case-analysis/
- Master System Design: https://eoepca.github.io/master-system-design/

Email: Eoepca.SystemTeam@telespazio.com

Questions

Any Questions ?

Our Questions

What data access interface should EP present to processor/apps?

Approach to federated authorisation (platform->platform)?

Approach to Catalogue federation?

Packaging/deployment of Interactive Applications?

What data interfaces to resource tier should be supported?

What standard interface for Catalogue CrUD operations?